# Time-Extended Policies in Multi-Agent Reinforcement Learning

Kagan Tumer
NASA Ames Research Center
Mailstop 269-4
Moffett Field, CA 94035
ktumer@mail.arc.nasa.gov

Adrian K. Agogino
NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035
adrian@email.arc.nasa.gov

## Abstract

*Reinforcement learning methods perform well in many domains where a single agent needs to take a sequence of actions to perform a task. These methods use sequences of single-time-step rewards to create a policy that tries to maximize a time-extended utility, which is a (possibly discounted) sum of these rewards. In this paper we build on our previous work showing how these methods can be extended to a multi-agent environment where each agent creates its own policy that works towards maximizing a time-extended global utility over all agents' actions. We show improved methods for creating time-extended utilities for the agents that are both "aligned" with the global utility and "learnable." We then show how to create single-time-step rewards while avoiding the pitfall of having rewards aligned with the global reward leading to utilities not aligned with the global utility. Finally, we apply these reward functions to the multi-agent Gridworld problem. We explicitly quantify a utility's learnability and alignment, and show that reinforcement learning agents using the prescribed reward functions successfully tradeoff learnability and alignment. As a result they outperform both global (e.g., "team games") and local (e.g., "perfectly learnable") reinforcement learning solutions by as much as an order of magnitude.*

## 1. Introduction

There are many problems which can only be properly addressed by having a set of autonomous agents act independently and have their *joint* sequence of actions maximize a pre-set global utility function. Examples of such problems include control of a constellation of satellites, construction of distributed algorithms, routing over a data network, and control of a collection of planetary exploration vehicles (e.g., rovers on Mars, or submersibles under Europa's ice caps). In such problems, agents have to solve two credit assignment problems at once. First, an agent has to figure out how an action taken now affects future rewards. This temporal credit assignment problem has been dealt with extensively in the single agent context; there are many reinforcement learning systems [10], (e.g., Q-learners [13]) that have successfully been applied to real world problems [1]. Second, an agent has to be able to choose actions that will, when combined with the actions of all other agents, lead to good values of the global utility. This structural credit assignment problem is difficult and is usually handled by having either each agent receiving the global utility as their private utility (e.g., "team" games [2]), or of imposing external mechanisms (e.g., contracts, auctions) that encourage the agents to work together [4, 8]. This paper addresses the structural credit assignment problem by designing private utilities for the agents that are both aligned with the global utility, and easier for the agents to learn to maximize. The agents will then use Q-learners to address the temporal credit assignment problem.

In earlier work, we discussed how time-extended utilities can be applied in a multi-agent system [11]. In this work, we extend that work by providing rewards whose undiscounted sums approximate those utilities, by explicitly computing the degree of alignedness between agent utilities and the global utility, and by computing the signal-to-noise properties of the derived utilities. We also provide a new utility that significantly outperforms the previous ones, especially in domains with hundreds of agents. In Section 2, we provide a summary how to make utilities that resolve the structural credit assignment problem. In Section 3 we discuss how to devise rewards that deal with the structural credit assignment problem for a single time step, while avoiding a common difficulty where using rewards aligned with the global reward leads to utilities not aligned with the global utility. In Section 4, we describe token collection in the Gridworld problem domain and develop agents' private utilities that allow agents using reinforcement learning to re-

solve both credit assignment problems simultaneously. In Section 5, we present simulation results that show that the utilities presented here possess high alignedness and learnability as compared to traditional approaches, and lead to solutions that significantly outperform those traditional approaches.

## 2. Factored and Learnable Utilities

In this work, we focus on multi-agent systems that aim to maximize a global utility function, $G(z)$, which is a function of the joint move of all agents in the system, $z$. Instead of maximizing $G(z)$ directly, each agent, $\eta$, tries to maximize its **private utility function** $g_\eta(z)$. Our goal is to create private utility functions that will cause the multi-agent system to produce high values of $G(z)$. Note that in many systems, an individual agent $\eta$ will only influence some of the components of $z$. We will use the notation $z_\eta$ to refer to the parts of $z$ that are dependent on the actions of $\eta$. The vector $z_\eta$ is the same size as $z$ and is equal to $z$ except that all the components that do not depend on $\eta$ are set to zero. Note that this subscripted vector notation is not the same as a traditional index to a vector since $z$ and $z_\eta$ have the same number of components.

There are two properties that are crucial to producing systems in which agents acting to optimize their own private utilities will also optimize the provided global utility. The first of these concerns "aligning" the private utilities of the agents with the global utility. Formally, a system is **fully factored** when for each agent $\eta$:

$$g_\eta(z) > g_\eta(z') \;\Leftrightarrow\; G(z) > G(z') \quad \forall z, z' \;\; s.t. \;\; z - z_\eta = z' - z'_\eta \quad .$$

Intuitively, for all pairs of states $z$ and $z'$ that differ only for agent $\eta$, a change in $\eta$'s state that increases its private utility cannot decrease the global utility. As a trivial example, any system in which all the private utility functions equal $G$ is fully factored [2]. In general though, one is more concerned with the degree of factoredness for a given utility function than for full factoredness. To address that concern, we define the degree of factoredness for a given utility function $g_\eta$ as:

$$\mathcal{F}_{g_\eta} = \frac{\int_z \int_{z'} u[(g_\eta(z) - g_\eta(z'))\ (G(z) - G(z'))] dz' dz}{\int_z \int_{z'} dz' dz} \quad (1)$$

where again, $z'$ denotes all states for which $z - z_\eta = z' - z'_\eta$ and $u[x]$ is the unit step function, equal to 1 if $x > 0$.

The second property, called **learnability**, measures the dependence of a utility on the actions of a particular agent as opposed to all the other agents. Formally we can quantify the learnability of utility $g_\eta$, in the vicinity of $z$ as the expected value over a new set of actions, $z'$, of $g_\eta$'s change in magnitude caused by the change in $\eta$'s action divided by $g_\eta$'s change in magnitude caused by the change in actions of all the other agents:

$$\lambda_{\eta, g_\eta}(z) = E_{z'} \left[ \frac{\|g_\eta(z) - g_\eta(z - z_\eta + z'_\eta)\|}{\|g_\eta(z) - g_\eta(z' - z'_\eta + z_\eta)\|} \right] \quad (2)$$

where $E[\cdot]$ is the expectation operator. So at a given state $z$, the higher the learnability, the more $g_\eta(z)$ depends on the move of agent $\eta$, i.e., the better the associated signal-to-noise ratio for $\eta$. Intuitively then, higher learnability means it is easier for $\eta$ to achieve a large values of its utility. Note in the factored team-game example above, the utility of each agent depended on the actions of all the agents. Such systems often suffer from low signal-to-noise, a problem that get progressively worse as the size of the system grows.

### 2.1. Designing Agent Utilities

Though the need for designing factored and learnable utilities for the agents is highlighted above, in general it is not possible for the utilities both to be factored and to have infinite learnability (i.e., no dependence of any $g_\eta$ on any agent other than $\eta$) for all of its agents [15]. However, consider a family of utility functions, called **Wonderful Life (WL)** utility functions. For each agent $\eta$, the WL utility is given by:

$$WLU^c_\eta = G(z) - G(z - z_\eta + c) \quad (3)$$

where $c$ is an arbitrary vector. For any choice of $c$, WL utilities have been shown to be factored [15]. Furthermore, it can be proven that in many circumstances, especially in large problems, $\lambda_{\eta, WLU}(z) \geq \lambda_{\eta, G}(z)$, i.e., WLU has higher learnability than does a team game [15]. This is mainly due to the second term of the WLU, which removes part of the effect of other agents (i.e., noise) from $\eta$'s utility (how much noise is removed depends on the domain). Though all WL utilities are factored regardless of the choice of $c$, the selection of $c$ affects their learnability. Therefore, in practice matching the proper value of $c$ to the domain greatly improves the performance of the system [15]. This paper will address how to handle the setting of $c$ in three separate ways: (1) setting $c$ to the zero vector, (2) setting $c$ to the expected value of $z_\eta$, and (3) taking the expected value of WLU over $c$.

The first method is to simply set $c$ to the zero vector allowing the WLU to be expressed as:

$$WLU^{c=\vec{0}}_\eta = G(z) - G(z - z_\eta) \quad (4)$$

In many circumstances this method is equivalent to removing that agent from the system, hence the name of this utility function. For such a $c$, WLU is closely related to the economics technique of "endogenizing a player's (agent's) externalities" and Vickrey tolls [12]. WLU also may appear to

have some similarities to Groves' mechanism [3] in mechanism design, though Groves' mechanism actually produces a team game by subtracting out a player's benefit already received from public goods.

The second method for setting $c$ is to set it to the expected value of $z_\eta$ instead of the zero vector. In this case the WLU can be expressed as:

$$WLU_\eta^{c=\overline{z_\eta}} = G(z) - G(z - z_\eta + \overline{z_\eta}) \quad (5)$$

where $\overline{z_\eta} = E[z_\eta | z_{-\eta}])$ gives the expected value for the action of $z_\eta$ given the actions of all other agents. This choice of $c$ usually results in higher learnability than the zero vector [15]. It is also often easy to compute and is still useful even if the expectation can only be approximated. Note that in addition to offering better learnability, the WLU has an additional advantage over the global utility: in many instances, its computation only requires partial (e.g., local) information. Indeed, either the information required for the computation of the global utility, or the global utility itself has to be broadcast which can put a heavy communication burden on the system, not to mention create a centralized, single point of failure. However WLU generally needs much less information because many components' impact cancel out (e.g., they are unchanged in both the first and second terms of the WLU) and therefore never need to be observed by an agent. In Section 4.2 we show this to be the case for the Gridworld problem.

The third method for addressing how to set $c$ is to take the expected value of the WLU over the values of $c$ that are possible actions of $\eta$. We call this utility the **Expected WL** Utility (EWU):

$$\begin{aligned} EWU_\eta &= E_{z'_\eta}[WLU_\eta^{c=z'_\eta}(z)|z - z_\eta] \\ &= E_{z'_\eta}[G(z) - G((z - z_\eta + z'_\eta)|z - z_\eta] \\ &= G(z) - E_{z'_\eta}[G(z - z_\eta + z'_\eta)|z - z_\eta] \quad .(6) \end{aligned}$$

Instead of setting $c$ to a fixed value, the EWU integrates over all values of $c$ that equal an action of $\eta$ (denoted $z'_\eta$). This computation theoretically results in higher learnability than the WL utility [15], though it is generally difficult to compute and often needs to be approximated.

## 3. Time-Extended Rewards

Often we face the task of creating a policy to determine a sequence of actions, which requires us to break down a time-extended utility into single-time-step rewards. Consider a system, where the global utility, $G$, is a function of a sequence of actions, $A$, of all the agents. ($A$ is an $\eta$ by $t$ matrix of actions. We will use $A_\eta$ to represent the actions of agent $\eta$ across all time, $A_t$ to represent the actions of all agents at time $t$, and $A_\eta, t$ to represent the actions of agent $\eta$ at time $t$. All such matrices have the same dimensionality as $A$, where the non-used elements are set to zero.) Also assume that the global utility is an undiscounted sum of global rewards (GRs): $G(A) = \sum_t GR_t(A)$. Since the global utility is a sum of rewards, we can attempt to maximize it by having each agent use a Sarsa learner[1], receiving a global reward at every time step. However, as discussed earlier, this approach (i.e., team game) suffers from poor learnability, particularly if there are many agents in the system.

Instead we want to use a WLU-based reward, and the direct approach, which we call the naive WLR is given by:

$$NWLR_{\eta,t}^{c_{\eta,t}}(A) = GR_t(A) - GR_t(A - A_{\eta,t} + c_{\eta,t}) \quad (7)$$

where $c_{\eta,t}$ is $\eta$'s a fixed action for time step $t$. This reward is factored at time step $t$, since $\eta$'s actions at time step $t$ cannot affect the value of the second term. However this reward is *not* factored *through* time (i.e., the sum of these rewards factored with GR does not produce a utility factored with the global utility). To illustrate this problem, consider a simple two-time-step, multi-agent problem where each agent can take one of two actions at each time step. For an agent $\eta$, we can draw a reward graph showing the outcome of its actions given the actions of the other agents. Potentially agent $\eta$ could have a different reward graph for each possible combination of actions of all of the other agents. Consider one of these reward graphs as illustrated in Figure 1 (top), showing how the global reward values depend on $\eta$'s actions (this graph actually comes for a Gridworld example problem discussed later in Section 4 though the problem details are not needed for the analysis here).

Figure 1 shows that if agent $\eta$ moves left on the first time step, the global reward will be ten on the first time step and zero on the second time step. If the agent moves right on the first time step the global reward will be zero for the first time step, but it will be either ten or fifteen on the second time step depending on whether its second action is left or right respectively. When the actions of all of the other agents are held constant, if agent $\eta$ is using a Sarsa learner with the global reward, it should form the policy of moving right for both time steps, maximizing the global utility: the sum of global rewards. However consider the values produced by the NWLR, shown in Figure 1 (bottom). If $c_{\eta,t}$ represents the go-right action, then the NWLR will equal GR, for the first time step, but will be different if it takes the go-right action on the first time step. Instead of ten or fifteen, the NWLR will evaluate to negative five or zero on the second time step. If the agent is using a Sarsa learner, we would expect the agent to take the left action at the first time step, which is sub-optimal with respect to the global utility. This mismatch of factoredness between rewards and utilities can be elucidated by focusing on the update rule for a simple

1 A Sarsa learner is used in this example for its simplicity. In the experiments performed in the results section, Q-learners are used instead.

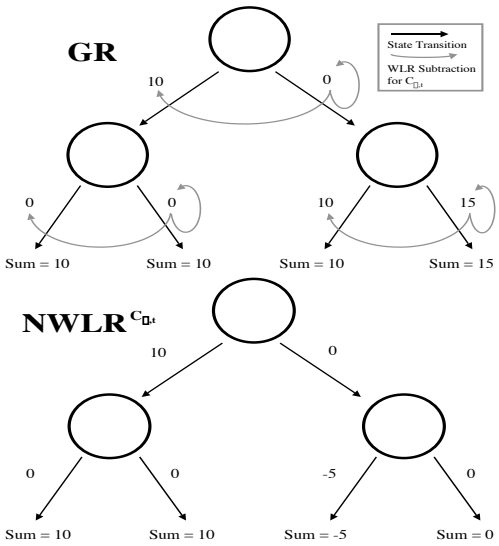


**Figure 1. Agent $\eta$ makes a choice of two actions for each of the two time steps. Values on the top graph represent the global rewards. Curved arrows show the values that are subtracted out to form the WL reward shown on the bottom graph. The NWLR is factored for a single time step since the NWLR and the GR result in the same ordering of values for each pair of actions at each time step. However, this form of WL reward is *not* factored through time since the NWLR and the GR give different orderings for the *sum* of rewards (i.e., the resulting utilities are not factored).**

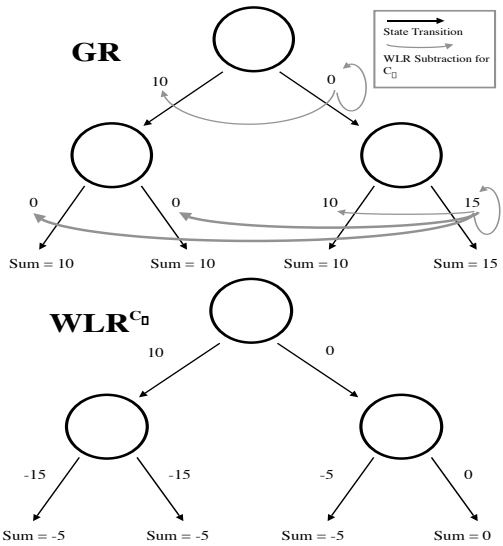


**Figure 2. Agent $\eta$ makes a choice of two actions for each of the two time steps. Values on the top graph represent the global rewards. Curved arrows show the values that are subtracted out to form the WL reward shown on the bottom graph. This form of WL reward is factored through time as the WLR and GR give the same ordering for the sum of rewards resulting form a sequence of actions.**

(deterministic and undiscounted) Sarsa learner:

$$\hat{Q}(s_t, a_t) = R_t + Q(s_{t+1}, a_{t+1}) \tag{8}$$

where the predicted sum of future rewards after taking action $a_t$ in state $s_t$ is calculated by adding the immediate rewards to the predicted sum of rewards for the next action. Using the $NWLR_{\eta,t}^{c_{\eta,t}}(A)$ in our example, the Sarsa update rule becomes:

$$\hat{Q}(s_t(A), a_t) = GR_t(A) - GR_t(A - A_{\eta,t} + c_{\eta,t}) + \hat{Q}(s_{t+1}(A), a_{t+1}) \tag{9}$$

The problem arises in $s_{t+1}(A)$ being dependent on the current action, $a_t$. That is not a problem in itself, but consider what $\hat{Q}(s_{t+1}(A), a_{t+1})$ is approximating:

$$\sum_{t'>t} GR_{t'}(A) - \sum_{t'>t} GR_{t'}(A - A_{\eta,t'} + c_{\eta,t}) \; . \tag{10}$$

The second sum is dependent on the current action $a_t$, causing the policy not to be factored. Even though the immediate reward is factored, the values in the Q-tables are not and the agents form a policy that is not aligned with the global utility.

The way to address this problem is to subtract out the full **sequence** of actions, $A_\eta$, instead of the single action, $A_{\eta,t}$, in the second term of the WLR and replace it with a constant full sequence of actions:

$$WLR_{\eta,t}^{c_\eta}(A) = GR_t(A) - GR_t(A - A_\eta + c_\eta) \tag{11}$$

where $c_\eta$ is a full sequence of actions. This version of WLR therefore differs from the previous version in that:

1. The subtracted single action $A_{\eta,t}$ is replaced with the *sequence* of actions $A_\eta$
2. The constant action $c_{\eta,t}$ is replaced with a constant *sequence* of actions $c_\eta$

For the example problem the values produced by this utility are shown in Figure 2 (bottom). In this case, $c$ is set to the "right-right" sequence of actions (i.e., 15 is subtracted from GR). Note that now the sum of rewards for both GR and WLR have the same ordering, that is the two utilities are factored. With this utility an agent using a Sarsa learner

forms the correct policy. In general though, this utility may be difficult to compute. It requires predicting the outcome of a sequence of rewards. In Section 4 we alleviate this problem by defining the a virtual "null" action for agents (e.g., $c_\eta$ is set to the zero vector), representing the agent being removed from the system.

## 4. Multi-agent Gridworld Problem

The single-agent Gridworld Problem [10] is a Markov Decision Process that is well known in the reinforcement learning community. In this problem, an agent navigates about a two-dimensional $n \times n$ grid, by moving a distance of one grid square in one of four directions: up, down, right or right. The state of an agent is the grid square it is on, and the reward an agent receives depends on the grid square it moves to. This paper uses an episodic, finite-horizon [5] model of the problem. In this model the agent starts at a start-state and then moves for a fixed number of time steps. At the beginning of each episode, the agent is returned to the start state. Reinforcement learners that maximize a sum of rewards, such as Q-learning, can be used in this problem.

To test our multi-agent utilities, we will use a multi-agent version of this well known Gridworld Problem. In the multi-agent version there are multiple agents navigating the grid simultaneously interacting with each others' rewards. This interaction is modeled through the use of tokens. Each token has a value between zero and one, and each grid square can have at most one token. When an agent moves into a grid square, the system receives a reward for the value of the token. The agent then removes the token so that a reward will no longer be received if it re-enters the same square or when another agent enters the grid square. If two agents move into the same square at the same time, it is only picked up once. The global objective of the Multi-agent Gridworld Problem is to collect the highest aggregated value of tokens in a fixed number of time steps.

We chose this problem because it is a standard problem in reinforcement learning research, and provides a clean testbed to compare the various utility functions. In the multi-agent version, the agent interactions provide a critical study of coordination and interference, as the agents have the potential to work at cross-purposes. Each agent attempting to maximize the value of the tokens it collects, can drive the global utility to severely sub-optimal values. As such, the design of the private utilities is crucial in this problem, and we address this issue below.

### 4.1. WLU and EWU for Gridworld

In this problem, the global utility is a function of the agents actions, $A$, the value of the tokens at each location, $\Theta$ ($\Theta_{x,y}$ giving the value of the token at location (x,y) ), and $L_0$, the initial locations of the agents. The global utility $G(A, \Theta, L_0)$ returns the value of the tokens received from a sequence of actions:

$$G(A, \Theta, L_0) = \sum_{x,y} \Theta_{x,y} I_{x,y}(A, L_0). \quad (12)$$

where $I_{x,y}(A, L_0)$ is an indicator function returning one if any agent entered the location $(x, y)$ and zero otherwise. The pseudo-code for $I_{x,y}(A, L_0)$ can be expressed as follows:

$I_{x,y}(A, L_0)$:
```
  for each η
    s ← L_0
    for each t
      if s_η = (x, y) return 1
      s ← δ(s, A_t)
  return 0
```

Since $\Theta$ and $L_0$ are always constant for a single experiment, $\Theta$ and $L_0$ will be omitted from any function parameter in the remainder of this paper to simplify notation (i.e., instead of using $z \equiv \Theta, L_0, A$, we will simply use $z \equiv A$).

Based on the definition and global utility given above, the EWU (given in Equation 6) becomes:

$$EWU_\eta(A) = G(A) - \sum_{A'_\eta} p_{A'_\eta} G(A - A_\eta + A'_\eta) \quad (13)$$

where the $A'_\eta$s are the possible action sequences agent $\eta$ can take. The second term in the equation is the expected value of the global utility over all the possible sequences of actions for agent $\eta$.

Now, let us formulate the WL utilities for this domain. First, setting $c_\eta$ to the zero matrix, we obtain WL utility where the agent is removed from the system[2]:

$$\begin{aligned} WLU^0_\eta(A) &= G(A) - G(A - A_\eta + c_\eta) \\ &= G(A) - G(A - A_\eta). \end{aligned} \quad (14)$$

This utility returns an agent's contribution to the global utility. Note, this utility differs from one where the values of the tokens present in the locations visited by the agent are summed (i.e., a utility based on an agent's immediate local effects). $WLU^0$ gives the value of the tokens *in locations not visited by other agents*, i.e., the values of token that would not have been picked up had agent $\eta$ not been in the system. This provides the marginal impact for that agent.

2 From here onward, we will refer to $WLU^{c=\vec{0}}$ as $WLU^0$ to simplify the notation.

Next, let us define the WL utility resulting from agent $\eta$ taking the virtual average action, where it partially takes all possible actions[3]:

$$WLU^a_\eta(z) = G(A) - G(A - A_\eta + A^{ave}_\eta) \quad (15)$$

where $A^{ave}_\eta$ is the average sequence of actions. Because in practice $A^{ave}_\eta$ can be hard to define, we discuss an alternative in the next section.

### 4.2. WL and EW Rewards for Gridworld

WLU and EWU are based on the performance over a full episode, and therefore are problematic to use directly in practice[4]. We therefore introduce single time step rewards whose undiscounted sums form these utilities. First, let us decompose an arbitrary utility $U$ in the following manner:

$$U(A) = \sum_t U(A_{<t+1}) - U(A_{<t}). \quad (16)$$

where $A_{<t} = \sum_{t'<t} A_{t'}$ is the action matrix representing all of the actions taken before time $t$. Now, it is possible to represent the single time step reward $R_t$ by:

$$R_t(A) = U(A_{<t+1}) - U(A_{<t}) \quad (17)$$

Now we can generate the four single-time-step reward versions of the four utilities[5]:

$$\begin{aligned}
GR_t(A) &= G(A_{<t+1}) - G(A_{<t}) && (18)\\
EWR_{\eta,t}(A) &= GR(A) \\
&\quad - \sum_{A'_\eta} p_{A'_\eta} GR_t(A - A_\eta + A'_\eta) && (19)\\
WLR^0_{\eta,t}(z) &= GR_t(A) - GR_t(A - A_\eta) && (20)\\
WLR^a_{\eta,t}(z) &= GR_t(A) \\
&\quad - GR_t(A - A_\eta + A^{ave}_\eta) && (21)
\end{aligned}$$

While we would like to use $EWR$ and $WLR^a$ precisely as formulated above, these rewards tend to be difficult to compute exactly. Since the set of action sequences grows exponentially with $t$, it is computational expensive to sum over all possible action sequences as needed to compute EWR and to compute the matrix $A^{ave}_\eta$ used in $WLR^a$. Instead we will approximate $EWR$ and $WLR^a$, using only the previous action instead of the entire sequence of actions:

$$\begin{aligned}
EWR_{\eta,t}(A) &= GR(A) \\
&\quad - \sum_{A'_{\eta,t}} p_{A'_{\eta,t}} GR_t(A - A_{\eta,t} + A'_{\eta,t}) \\
WLR^a_{\eta,t}(z) &= GR_t(A) - GR_t(A - A_{\eta,t} + A^{ave}_{\eta,t})
\end{aligned}$$

where $A^{ave}_{\eta,t}$ is the average single action. This is a virtual action defined as going in all four directions at once. Note that these utilities are no longer factored through time since only $\eta$'s action for a single time step is subtracted out. However we will show in Section 5 that in practice these utilities are very close to being factored and are very effective.

Directly computing the utilities represented in Equations 18-21 requires knowledge about the state of the entire system. However, to compute $WLR^0$ an agent only needs to observe all the squares to which it has been. All the other terms cancel out. In a domain such as Mars Rovers this could be done by laying a trail of sensors along its path. The utilities $EWR$ and $WLR^a$ have similar requirements except they have to observe a few squares around every square they have been. Again in the Mars Rover domain this could be done by laying a trail of sensors that have a small radius of observation. The only utility that truly requires full observability is the team game utility, $G$.

## 5. Experimental Results

To evaluate the effectiveness of the collective-based approach in the Multi-agent Gridworld, we conducted experiments in token worlds with 10, 85 and 200 agents. The token worlds had a similar distribution of tokens, where the "highly valued" tokens were concentrated in one corner, with a second concentration near the center where the rovers were initially located. For an $nxn$ grid the value of a token in position $(x, y)$ was $(x + y)/n - 1$ when $(x + y)/n$ was greater than 0.4. In addition the tokens at locations $(m/2, m/2 - 1)$ and $(m/2 + 1, m/2 - 1)$ were set to 0.8. All other tokens had a value of zero. Agents start an episode at location $(m/2, m/2)$. To keep the approximate difficulty of the token collection problem constant with respect to the number of agents, the ratio of the number of grid squares to number of agents was held constant. The size of the token world was 10x10 for ten rovers, 29x29 for 85 agents (e.g., 8.4 times larger than for 10 agents), and 44x44 for 200 agents (e.g., 20 times larger than for 10 agents). In all the experiments, the agents used Q-learners to learn their policy (we expect a Sarsa learner to produce similar results). Each run consisted of 1000 episodes of 10, 29 and 44 steps respectively for the the 10, 85 and 200 agent systems. There were 100 runs per each 10 and 85 agent experiment (e.g., for 10 agents, we had 100 runs of 1000 episodes of 10 time steps) and 24 runs per each 200 agent experiment. The discount rate was 0.95 and the learning rate was set to $1/(1+0.0002*v_{s,a})$ where $v_{s,a}$ is the number of times an agent took action $a$ in state $s$. Given the Q-values, the action were chosen with Boltzmann selector with $k = 50$ and tables were initially set to zero as traditionally done to trade-off exploration vs. exploitation [10]. Table 1 shows the results of a ten rover system for the five utilities.

3 To simplify notation in what follows we refer to $WLU^{c=A^{ave}_\eta}$ (utility obtained by setting $c$ to the average action sequence of $\eta$ as $WLU^a$.

4 Providing a reward of zero in all but the final step where the full utility is given as a reward is an undesirable solution because it makes filling the Q-table an impossibly difficult task.

5 In the actual implementation there are some tie breaking rules if more than one agent goes into the same square at the same time.

| Agent Utitlity | Normalized World Utility | Deviation in Mean | Convergence Time |
|---|---|---|---|
| $WLU^a$ | 0.998 | 0.001 | 40 |
| $WLU^0$ | 0.993 | 0.002 | 70 |
| $EWU$ | 0.97 | 0.002 | 110 |
| $G$ | 0.37 | 0.02 | 770 |
| $PLU$ | 0.29 | 0.02 | 10 |
| (Random | 0.34 | 0.1) | |

**Table 1. Gridworld Performance for 10 Agents**

| Agent Utility | Factoredness (in %) | Learnability |
|---|---|---|
| $WLU^a$ | $99.0 \pm 0.078$ | $1.78 \pm 0.03$ |
| $WLU^0$ | $100.0 \pm 0.0$ | $0.96 \pm 0.03$ |
| $EWU$ | $99.3 \pm 0.088$ | $1.15 \pm 0.03$ |
| $G$ | $100.0 \pm 0.0$ | $0.2 \pm 0.0026$ |
| $PLU$ | $86.2 \pm 0.66$ | $\infty$ |

**Table 2. Factoredness and Learnability Estimates for 10 Agents**

The performance of five private utility functions was tested: (i) the Perfectly Learnable Utility (PLU), where each agent receives the weighted total of the tokens that it alone collected. It is the natural extension of the single agent problem, and represents the optimal utility in the single rover domain. The PLU is a function of the moves of only a single agent and therefore as infinite learnability, but is not generally factored. (ii) the Team Game (TG) utility, where each agent received the full global utility. It is the opposite extreme of the PLU since it is fully factored, but has very low learnability. (iii) the $WL^0$ utility, where $c$ is set to zero. Intuitively, this utility computes the contribution an agent makes to the token collection, by looking at the difference in the total token collection with and without that agent. (iv) the $WL^a$ utility, where $c$ is set to $A^{ave}$, representing the difference between the utility value resulting from an agent's actual action and its "smeared" action; and (v) the EWU, where the agent's contribution is computed as the difference between the action it took and its expected action.

The performance measure in these figures is "normalized" global utility given by $\frac{G(A)}{\sum_{x,y}\Theta_{x,y}}$. This normalized utility provides the fraction of token values that was collected by the agents (a value of one means all available tokens were collected). The deviation in the mean gives the $\frac{\sigma}{\sqrt{N}}$ where $N$ is the number of runs ($N = 100$ for this experiment). The convergence times is the time taken to reach $.9(U_{max} - U_{rand}) + U_{rand}$. The results show that PLU produced poor results, results that were indeed worse than random actions. This is caused by all agents aiming to acquire the most valuable tokens, and congregating towards the corner and center of the world where such tokens are located. In this case agents using the PLU competed, rather than cooperated with one another. Note however, that the convergence time was extremely rapid. The agents using TG fared marginally better, but their learning was slow. This system was plagued by the signal-to-noise problem associated with each agent receiving the full global reward for each individual action they took. In contrast, agents using $WL^0$ and EWU performed very well, and agents using $WL^a$ performed almost optimally. In each of these three cases, the reinforcement signal the agents received was both factored and showed how their actions affected the global reward more clearly than did the TG reinforcement signal.

We can gain some insight into these results by calculating the learnability and factoredness of these utilities, using Monte-Carlo methods. Factoredness is computed by taking the action, $A_{\eta,t}$ for an agent $\eta$ at a random time $t$ and replacing it with a random action $A'_{\eta,t}$. If $n$ samples are taken, an estimate for the degree of factoredness introduced in Equation 1 for a private utility $g$ is given by:

$$\frac{1}{n_d}\sum_{i=1}^{n} \quad u[(g(A) - g(A - A_{\eta,t} + R^i_{\eta,t})) \\ (G(A) - G(A - A_{\eta,t} + R^i_{\eta,t}))]$$

where $u$ is the unit step function (output of 1 if argument is strictly greater than zero), $R^i_{\eta,t}$ is the $i$th random action for agent $\eta$, and $n_d$ is the number of times the random action caused a change in $G$. Similarly learnability can be approximated as:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\|U(A) - U(A - A_{\eta,t} + R^i_{\eta})\|}{\|U(A) - U(R^i - R^i_{\eta,t} + A^i_{\eta})\|}$$

where and $R^i - R^i_{\eta,t}$ is the $i$th random action for all agents ther than $\eta$.

Table 2 shows the factoredness and learnability estimates respectively, along with the differences in the mean for each. The results indicate that all of the utilities except for $PLU$ are either perfectly factored or very close to being factored. Instead the improved performance of $WLU^a$ can be attributed to it having higher learnability than either $EWU$ or $WLU^0$, a result consistent with the convergence times reported in table 1.

Tables 3 and 4 show the experimental results for the larger systems (85 and 200 agents). The results are qualitatively similar to those with 10 agents, though the differences become more pronounced. The team game agents have a harder time learning, and perform randomly for the 200 agent case. Furthermore, the performance of $WL^a$ is now clearly superior to that of $WL^0$, showing that using the

| Agent Utility | Normalized World Utility | Deviation in Mean |
|---|---|---|
| $WLU^a$ | 0.84 | 0.007 |
| $WLU^0$ | 0.71 | 0.008 |
| $EWU$ | 0.73 | 0.009 |
| $G$ | 0.06 | 0.006 |
| $PLU$ | 0.018 | 0.0006 |
| (Random | 0.058 | 0.02) |

**Table 3. Gridworld Performance for 85 Agents**

| Agent Utility | Normalized World Utility | Deviation in Mean |
|---|---|---|
| $WLU^a$ | 0.57 | 0.01 |
| $WLU^0$ | 0.38 | 0.01 |
| $EWU$ | 0.41 | 0.01 |
| $G$ | 0.025 | 0.005 |
| $PLU$ | 0.007 | 0.0002 |
| (Random | 0.02 | 0.02) |

**Table 4. Gridworld Performance for 200 Agents**

degree of freedom of being able to use an arbitrary $c_\eta$ provides significant improvements over solutions aimed at "endogenizing externalities" ($WL^0$).

## 6. Discussion

In this article we extended previous work on designing distributed reinforcement learning algorithms for multi-agent systems. We decomposed agent utility functions requiring sequences of actions into single step rewards that conserve the salient features of the agent utilities. Furthermore, we exposed the dangers of some single step rewards aligned with a global reward leading to utilities which are not aligned with the global utility. Our analysis shows the simple steps required to overcome this problem. Our experimental results were conducted in a gridworld scenario, a problem related to many real world problems including exploration vehicles trying to maximize aggregate scientific data collection (e.g., rovers on the surface of Mars). Furthermore, we computed the factoredness and learnability for the different agent utility functions using Monte-Carlo methods. These results shed light on the performance and convergence times of the different utilities and validated the assumptions made in the derivation of the utilities.

The results demonstrate that agent utilities designed to have both high factoredness and high learnability outperform both "perfectly learnable" utilities and fully factored utilities based on "team games" (e.g., using global utility). Even the simplest of our utilities, $WL^0$, showed marked improvement over such utilities, while $WL^a$ showed further improvements. The factoredness and learnability estimates showed that WL-based utilities had high factoredness and high learnability, thus using the best features of both team games and PLUs. Our current research consists of extending these results to domains with partial observability.

## References


[1] J. A. Boyan and M. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems - 6*, pages 671–678. Morgan Kaufman, 1994.

[2] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In Touretzky, Mozer, and Hasselmo, eds, *Advances in Neural Information Processing Systems - 8*, pages 1017–1023. MIT Press, 1996.

[3] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[4] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, June 1998.

[5] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[6] M. Kearns and D. Koller. Efficient reinforcement learning in factored MDPs. In *Proceedings of the 16th Intl. Joint Conf. on Artificial Intelligence*, pages 740–747, 1999.

[7] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

[8] T. Sandholm and R. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37:147–166, 1995.

[9] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3), 2000.

[10] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[11] K. Tumer, A. Agogino, and D. Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Proc. of the First Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pages 378–385, Bologna, Italy, July 2002.

[12] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[13] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3/4):279–292, 1992.

[14] D. Wolpert and J. Lawson. Designing agent collectives for systems with markovian dynamics. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.

[15] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.